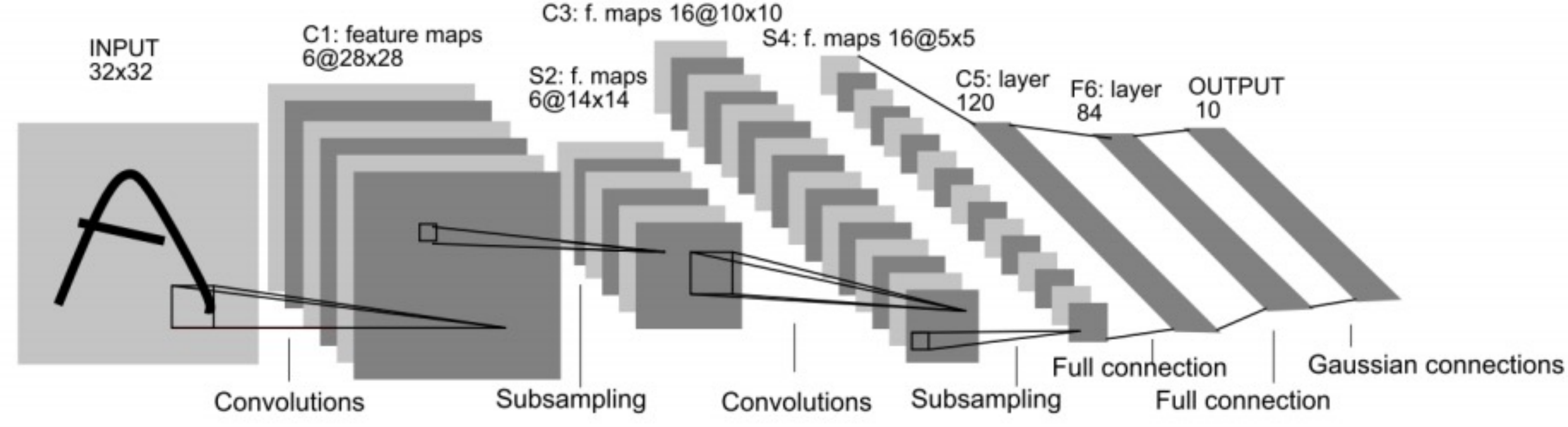
Computer Vision

Convolutional Neural Networks
Architectures

Luis F. Teixeira | M.EIC | FEUP

LeNet 5, LeCun 1998



- Input: 32x32 pixel image. Largest character is 20x20 (All important info should be in the center of the receptive field of the highest level feature detectors)
- Cx: Convolutional layer
- Sx: Subsample layer
- Fx: Fully connected layer

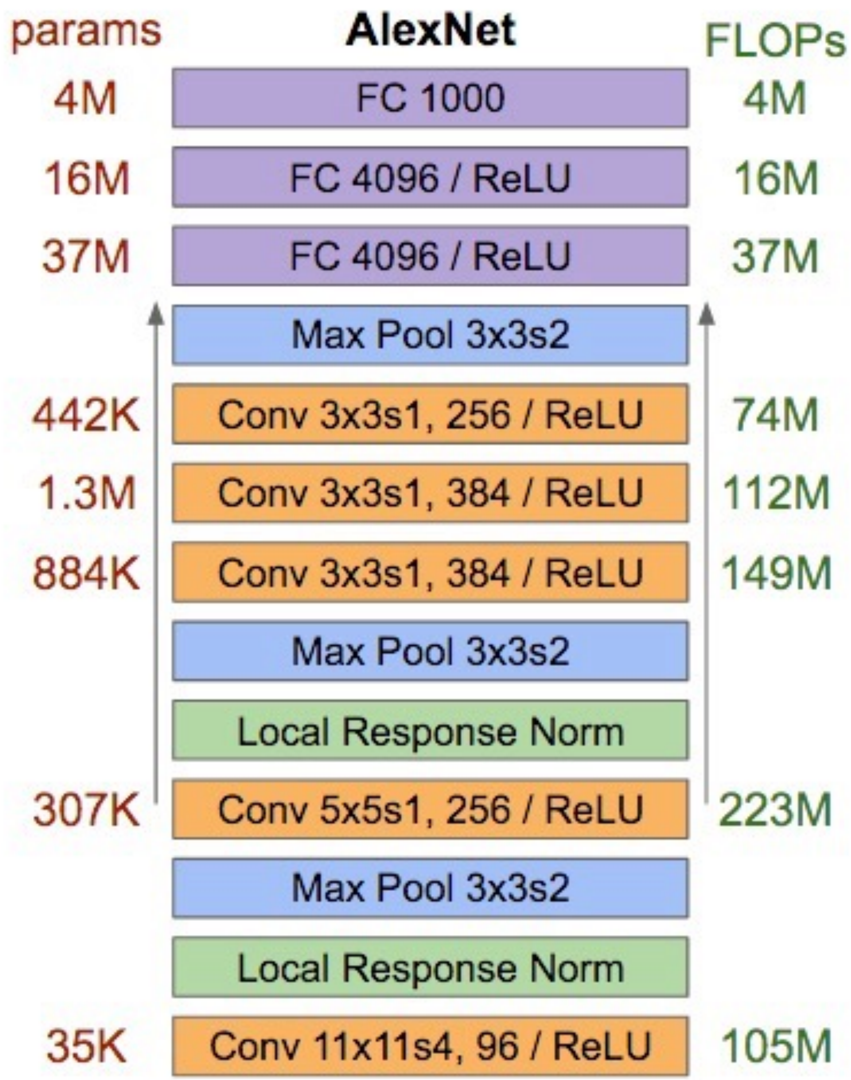
LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., & others. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278–2324

AlexNet

Layers defined by:
- Kernel sizes
- Strides
- # channels
- # kernels
- Pooling

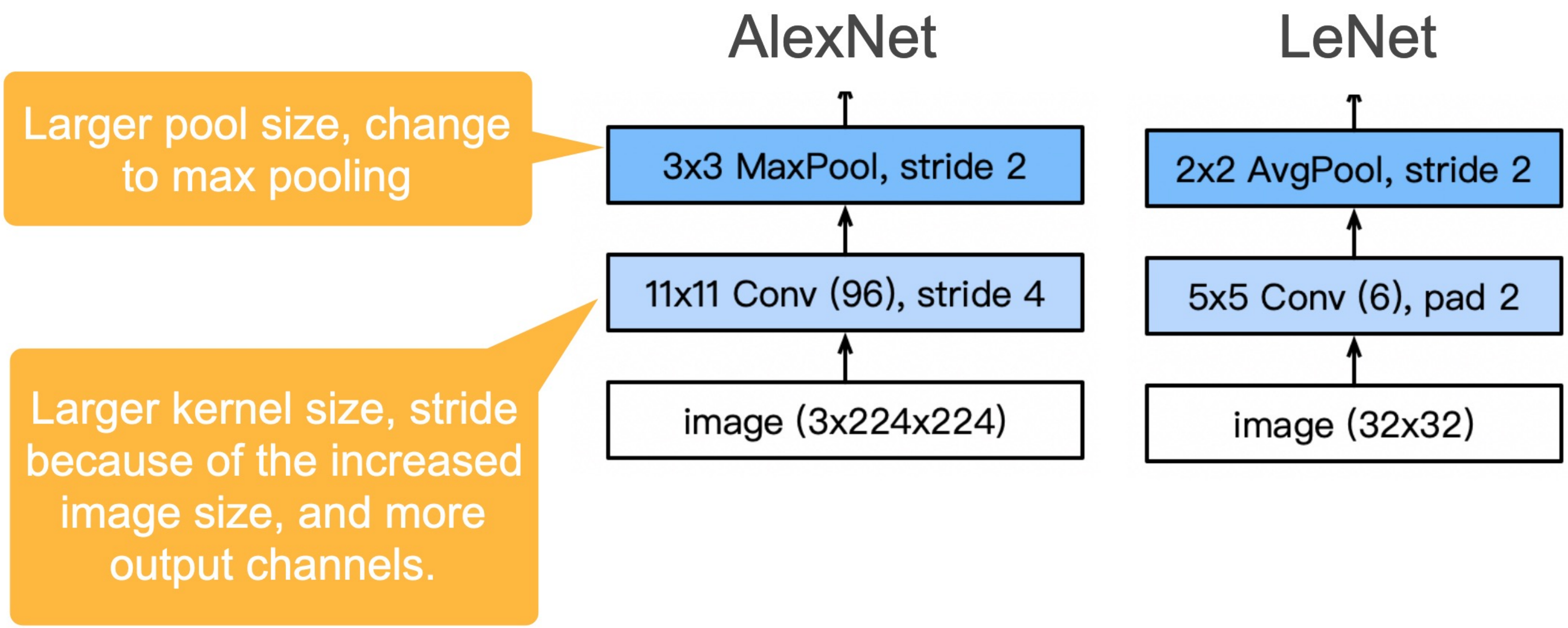
Total parameters: 61M

vs. LeNet
Total parameters: 60k



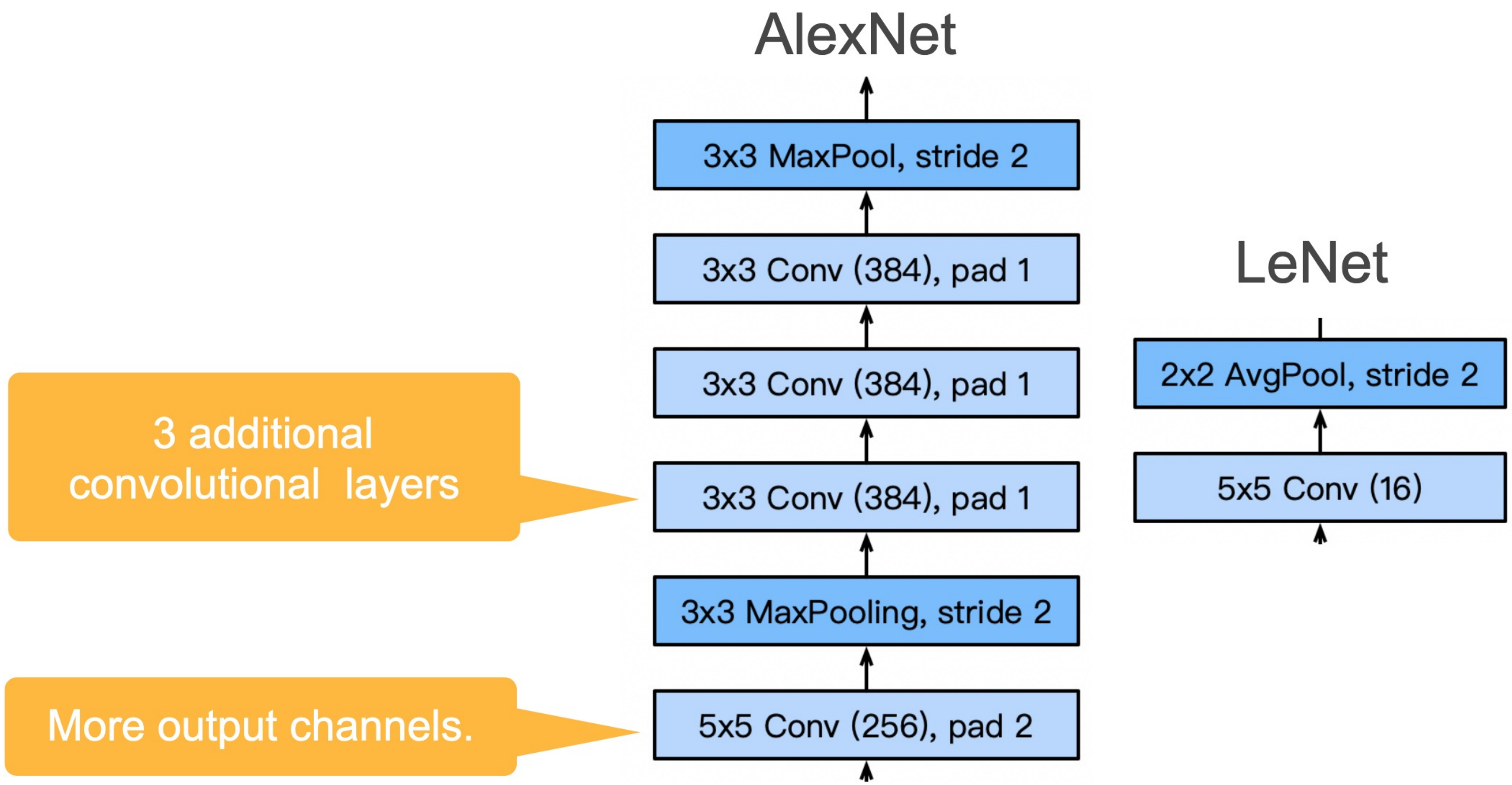
Krizhevsky, Alex; Sutskever, Ilya; Hinton, Geoffrey E. "ImageNet classification with deep convolutional neural networks" . Communications of the ACM. 60 (6): 84–90.

AlexNet



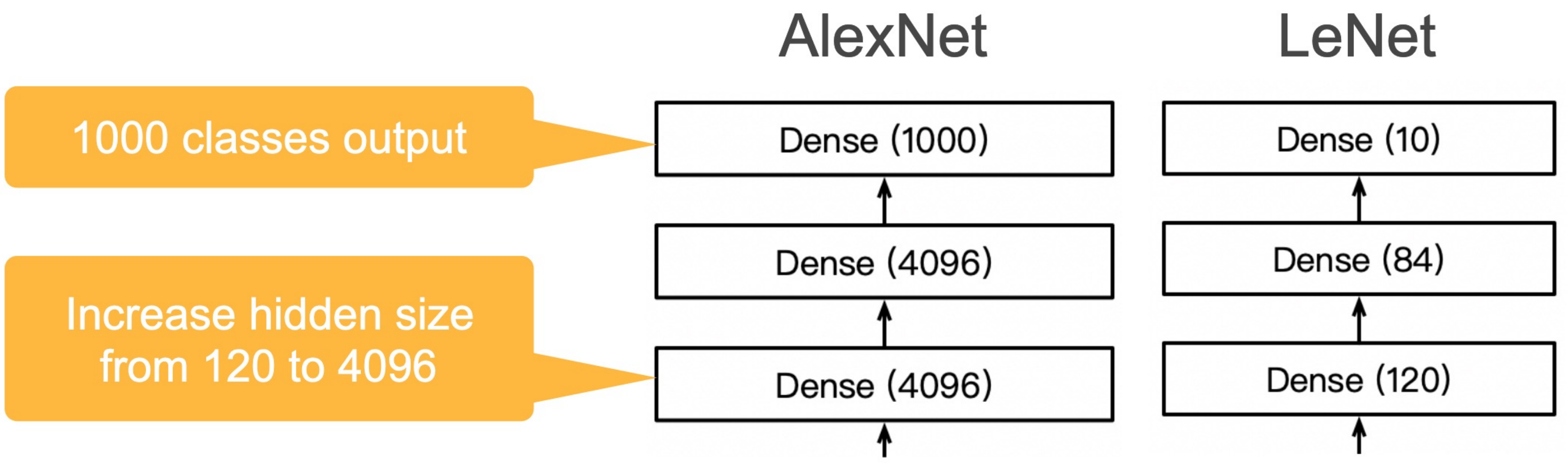
Larger pool size, change to max pooling
Larger kernel size, stride because of the increased image size, and more output channels.
AlexNet
LeNet
3x3 MaxPool, stride 2
11x11 Conv (96), stride 4
image (3x224x224)
2x2 AvgPool, stride 2
5x5 Conv (6), pad 2
image (32x32)

AlexNet



AlexNet
3x3 MaxPool, stride 2
3x3 Conv (384), pad 1
3x3 Conv (384), pad 1
3x3 Conv (384), pad 1
3x3 MaxPooling, stride 2
5x5 Conv (256), pad 2
LeNet
2x2 AvgPool, stride 2
5x5 Conv (16)
3 additional convolutional layers
More output channels.

AlexNet

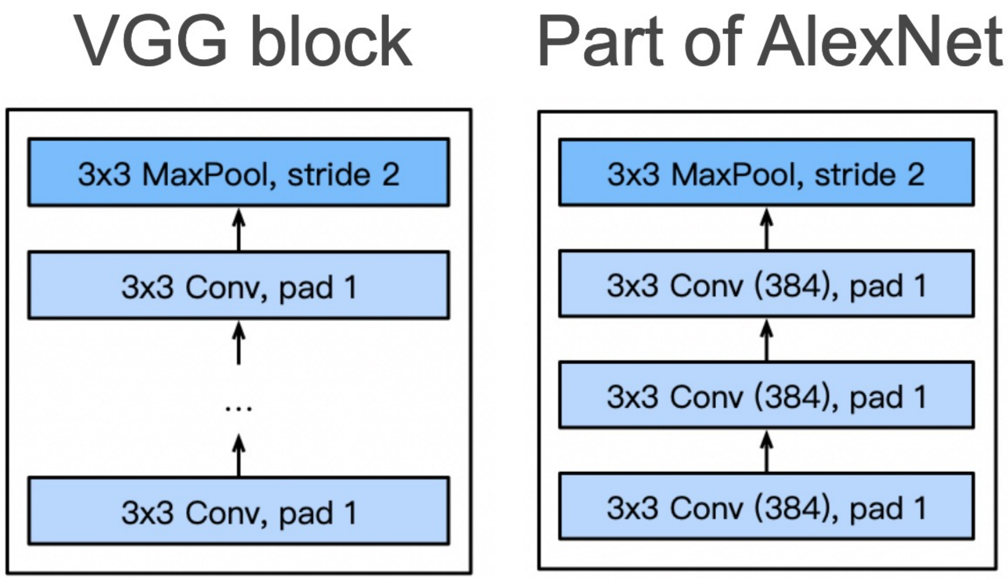


More tricks:

- Change activation function from sigmoid to ReLU (reduce the vanishing gradient problem)
- Add a dropout layer after two hidden dense layers (better robustness / regularization)
- Data augmentation
- GPUs

VGG

- Deeper vs. wider?
  - 7x7 convolutions
  - 3x3 convolutions (more)
  - **Deep & narrow better**
- VGG block
  - 3x3 convolutions (pad 1) (n layers, m channels)
  - 2x2 max-pooling (stride 2)
- 138M parameters

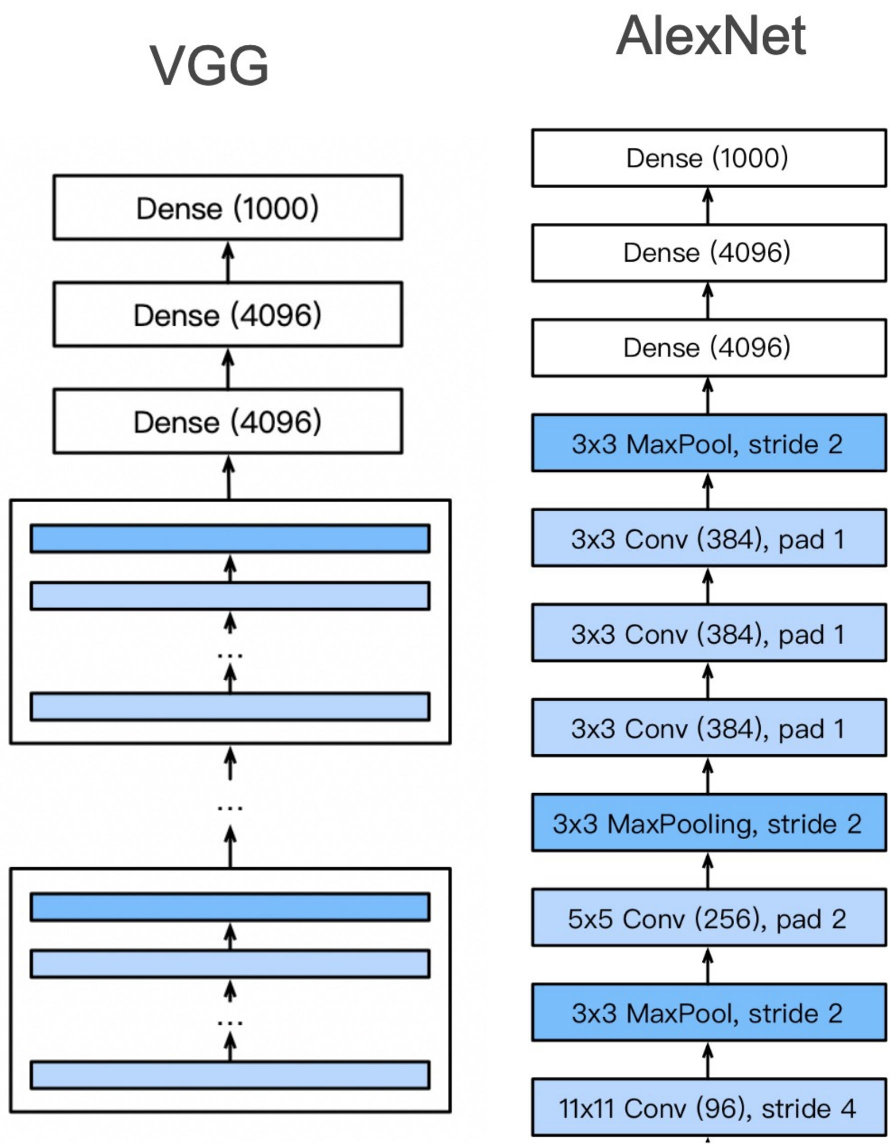


Note:
- 2 stacked layers of 3x3 cover an area of 5x5
- 3 stacked layers of 3x3 cover 7x7
- Number of parameters is reduced

K. Simonyan, A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition International Conference on Learning Representations, 2015

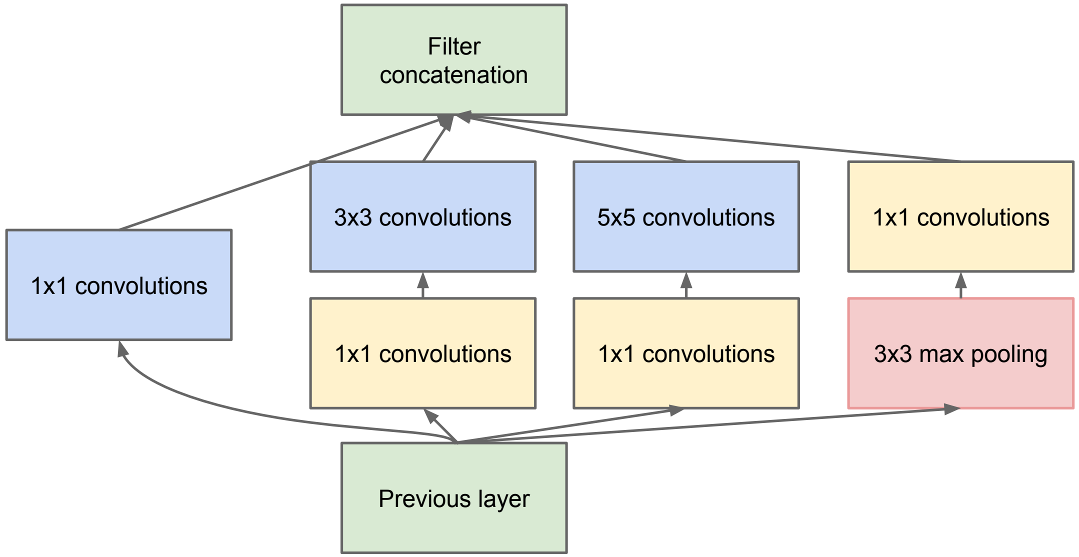
VGG

- Multiple VGG blocks followed by dense layers
  - Increasing number of filters 64-128-256-512
  - Decreasing image size 224-112-56-28-14-7
- Vary the repeating number to get different architectures, such as VGG-16, VGG-19, ...



Inception

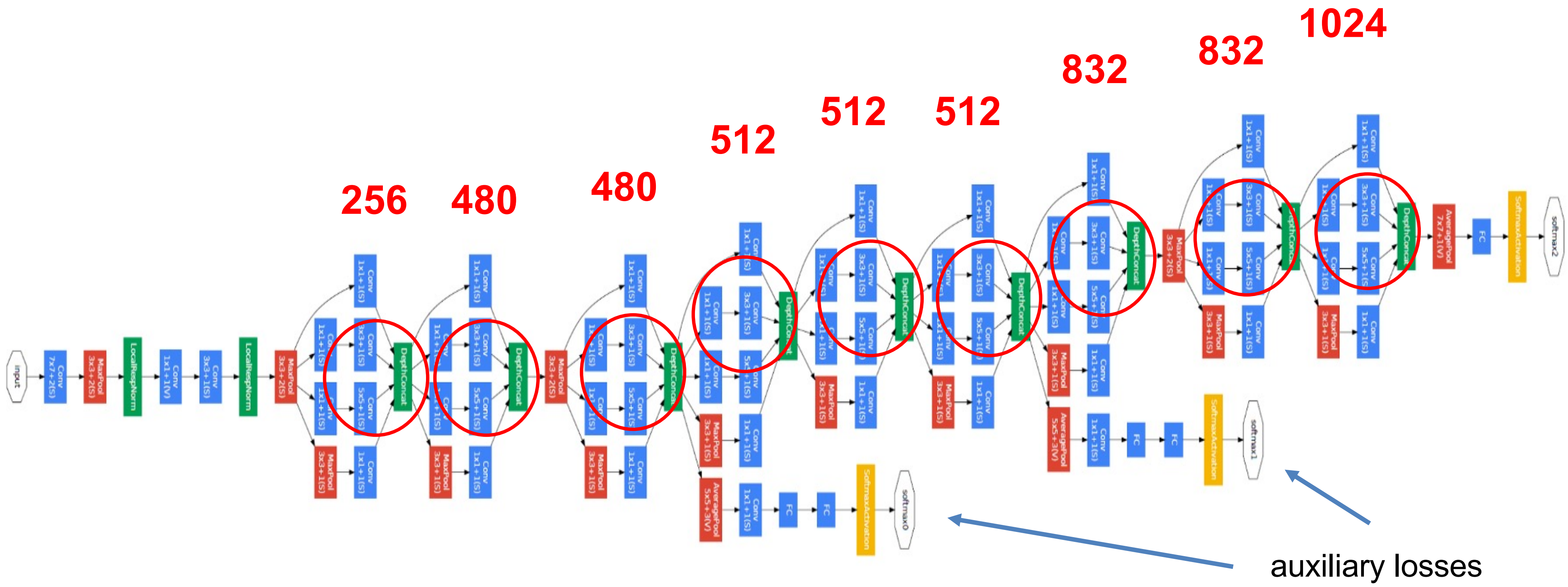
- GoogLeNet **Inception** module
- **Multi-scale** convolutional module
- Objective of reducing computational cost but preserving accuracy



Szegedy, Christian,et al. "Going deeper with convolutions." Proceedings of the IEEE Conference on CVPR. 2015

Rob Fergus

Inception

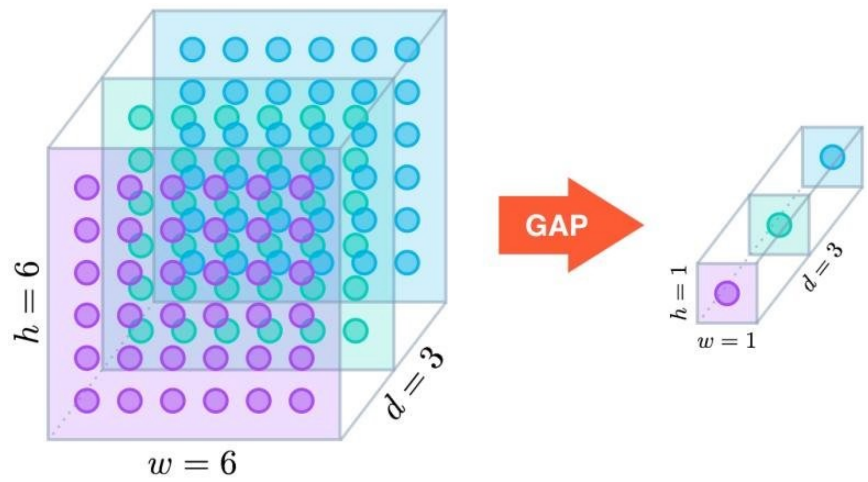


- Width of inception modules ranges from 256 filters (in early modules) to 1024 in top inception modules.
- Number of parameters: 5 million (much less than VGG)
- There were successive versions of Inception (v1, v2, v3 and v4) looking for more efficiency
- Output uses **Global Average Pooling**

Rob Fergus

Global Average Pooling

- Typically used as one of the last layers
- Reduces the need for (or totally replaces) the FC layers



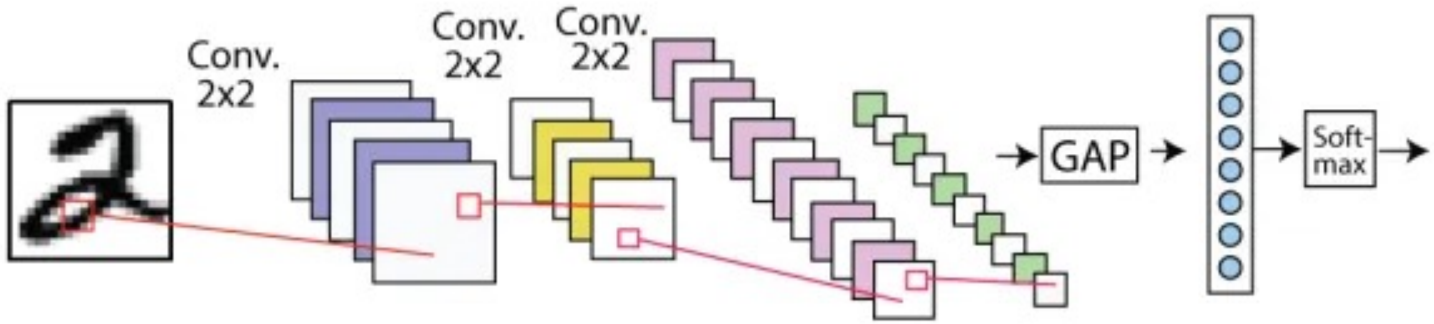
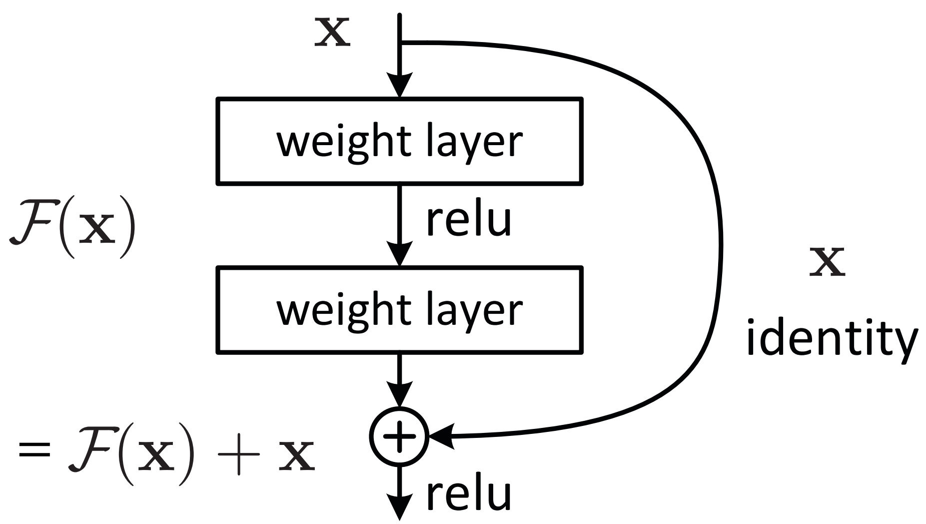
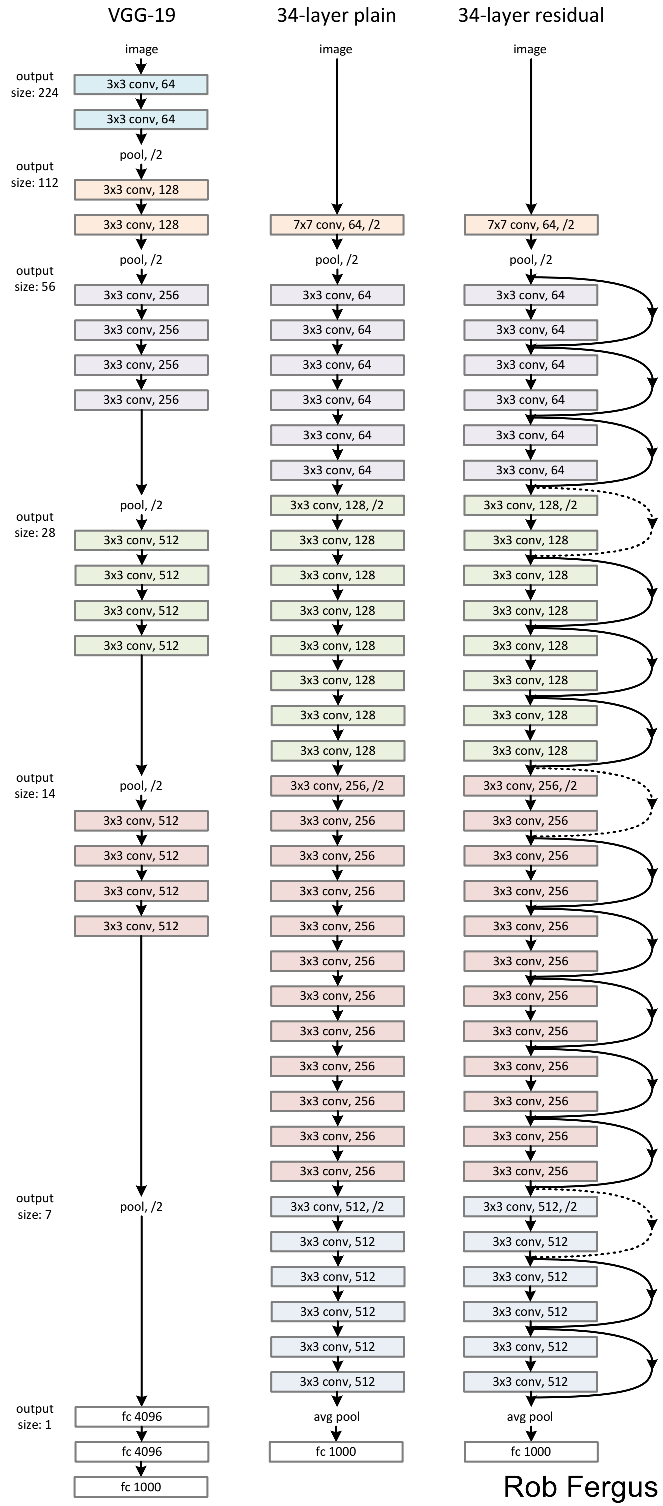


Image 1 source: https://www.researchgate.net/figure/Global-Average-Pooling_fig3_337550186
Image 2 source: https://principlesofdeeplearning.com/index.php/a-tutorial-on-global-average-pooling/

ResNet

- Really deep CNNs don't train well
- Key idea: introduce "pass through" into each layer (**skip connections**)
- Gradient can backpropagate more easily
- Learning **residuals** instead of full mapping → $\mathcal{F}(\mathrm{x}) = y - x$

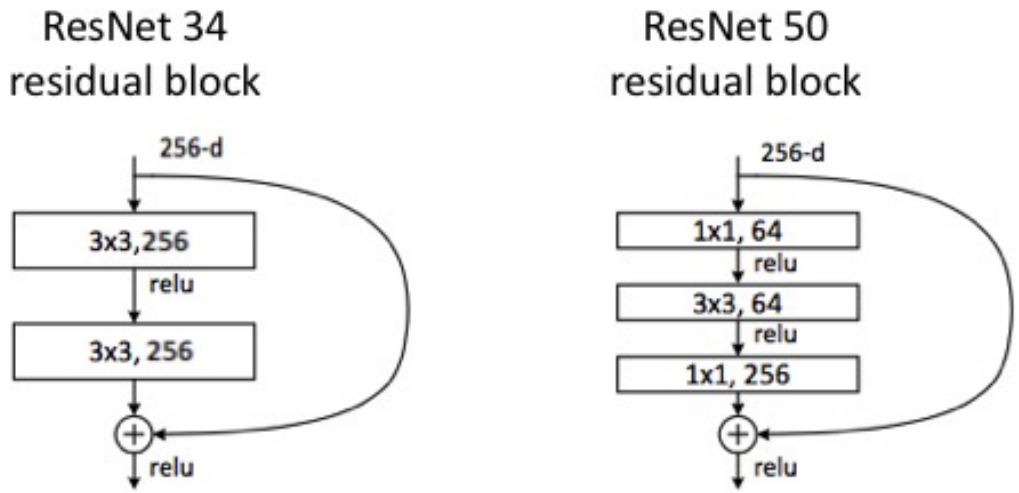




Rob Fergus

ResNet

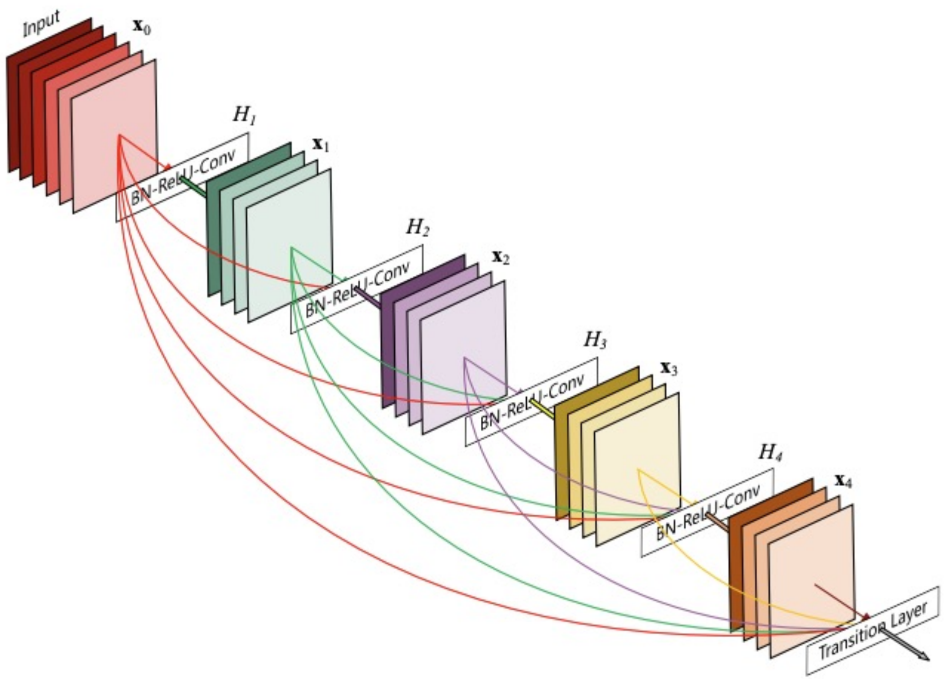
- Another version to ResNet made each residual module more efficient with Inception ideas of decomposing convolutions



- # parameters:
  - ResNet 34 – 21.3M
  - ResNet 50 – 23.5M

DenseNet

- **DenseNet** is an extension to ResNet
- Instead of adding the input to the mapping function, DenseNet **concatenates** them, so any previous learned features can be **reused** later
- Less new channels are needed in every layer
- Better performance with less complexity

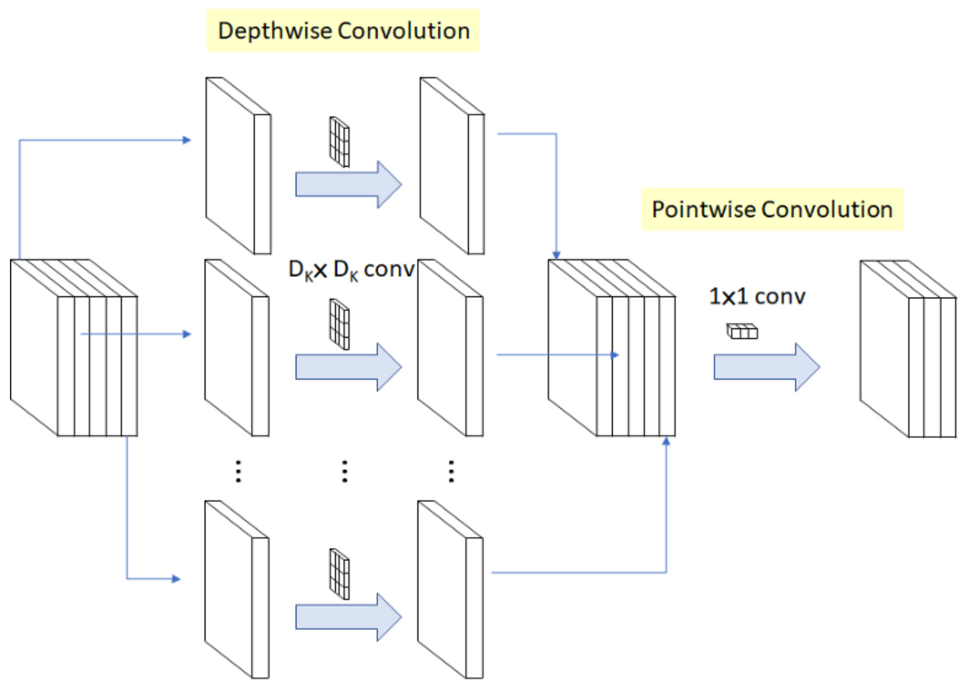


adapted from José Luis Alba

MobileNet

- These models can be **computationally expensive** → How can real-time performance be achieved using mobile or other embedded devices?

- **MobileNet** designed for memory and computation restrictions
  - Depth-separable convolutions
  - Reduce the number of parameters and computation



- **MobileNetV2** introduced tweaks like residual connections to reduce even more the parameters

Image Source: https://towardsdatascience.com/review-mobilenetv1-depthwise-separable-convolution-light-weight-model-a382df364b69 adapted from José Luis Alba

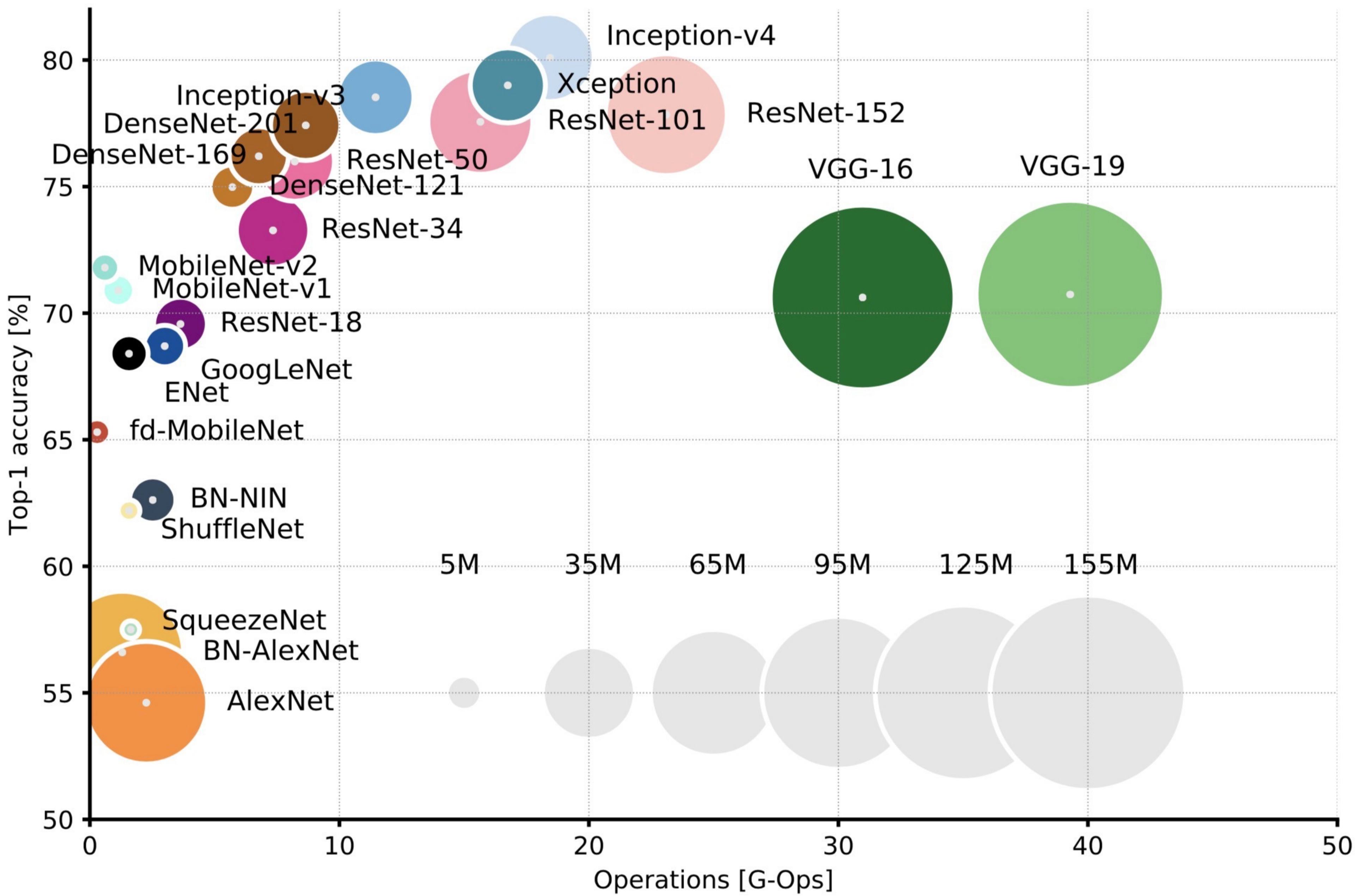


Image Source (2017): https://towardsdatascience.com/neural-network-architectures-156e5bad51ba